Django Deploy no Heroku

https://django.school

Francisco André

Sumário

Agradecimentos

Antes de começarmos preciso fazer um agradecimento especial a um amigo designer, profissional de excelente qualidade, criativo e super competente. Este teve a humildade de contribuir neste projeto logo no primeiro pedido e estou imensamente grato por sua ajuda.

A capa desse e-book é mais uma de suas criações, **Erie Dias**, aceite meus sinceros agradecimentos pela sua ajuda e excelente ilustração.

Se você precisa de um designer eu super indico este profissional, assino embaixo pelo seu comprometimento e qualidade.

Segue seus dados de contato:

Erie Dias – Designer Gráfico

http://behance.net/eriedias

eriedias7@gmail.com

+55 (86) 9 9819-3129

Introdução

Todo programador anseia após longos dias de desenvolvimento, mostrar sua criação, sua obra de arte ao mundo, mas alguns temem essa etapa, para outros é tranquilo, *no problem*, para outros essa é só mais uma etapa, enfim, uma coisa é certa, todos vão precisar colocar no ar sua aplicação, se não fizerem isso, de nada valeu todo o esforço para o desenvolvimento dela.

Como para alguns, encarar uma tela preta e meter a mão na massa configurando vários serviços necessários para que uma aplicação possa funcionar pelo menos com o mínimo possível é uma coisa do outro mundo, nasceu assim o Heroku, você continua na tela preta, mas agora o trabalho pesado é com eles, você simplesmente atende às suas exigências e com um comando, cria toda a estrutura necessária para sua aplicação.

Com a popularização da Cloud Computing (Computação em Nuvem), surgiram vários serviços e um deles é o alvo desse e-book, no caso, PaaS, alguns com certeza nunca ouviram falar por aí, outros já ouviram e sabem muito bem do que se trata, no entanto, para os que nunca ouviram falar, vou tentar de uma vez por todas esclarecer aqui o significado desse termo, vamos focar apenas neste termo, mas existem vários termos e serviços no mercado que fazem uso ou estão relacionados com Cloud Computing.

PaaS - Platform as a Service: um serviço focado no desenvolvedor, em um PaaS é possível fazer coisas como programar, compilar, *debugar* e deploy de uma aplicação. Nesse e-book vamos utilizar um PaaS somente para o Deploy.

Exemplo: Heroku (alvo desse e-book ;-)), Repl.it

Agora sim, tudo explicado e passado a limpo, vamos ao que interessa. Bom, o objetivo desse e-book é demonstrar em detalhes todo o processo de Deploy de uma aplicação Django no Heroku, um PaaS extremamente popular, de fácil manuseio e grátis até um limite de horas mensais, você vai saber mais sobre o plano free em breve.

Um pouco mais sobre as tecnologias que vamos usar

Se você está lendo esse livro, com certeza você sabe o que é o Django ou pelo menos tem ideia do que seja, mas para esclarecer eventuais dúvidas, irei nesse momento dá uma pincelada nesse assunto.

O Django é um *framework web* escrito em Python, uma linguagem de propósito geral, bastante versátil e multiplataforma, isso quer dizer que os programas desenvolvidos em Python rodam em qualquer lugar. E para deixar a coisa ainda mais clara (peço desculpa a galera que já saca disso, mas acho importante essa parte introdutória pois dar oportunidade aos que estão de fato iniciando do zero.), Framework é um software que ajuda os programadores a serem mais produtivos e a criarem programas mais fáceis de escalar e mais organizados, permitindo assim que outros programadores, não importando sua localização física ou mesmo idioma, consiga trabalhar em conjunto ou dá continuidade no trabalho com esse mesmo software/framework.

Sobre o Heroku, o Heroku é um PaaS que surgiu em meados de 2007 suportando inicialmente apenas Ruby, logo depois se tornou uma plataforma poliglota, suportando outras linguagens como Python, Java, JavaScript, PHP, Clojure, Go e Scala. A história da plataforma é bastante interessante, tem uma página na Wikipedia que fala mais a respeito, vale a pena uma passada rápida por lá.

Sobre a aplicação

Em um curso criado também por mim e disponibilizado pela Udemy, eu desenvolvi uma aplicação de To-Do List, esse curso vai do básico de Python até o Deploy da aplicação desenvolvida em Django no Heroku. Neste e-book usaremos a mesma aplicação, visto que o objetivo é demonstrar o processo de Deploy, então para não alongar muito o e-book e evitar a fadiga, vamos fazer uso da mesma, se você quiser praticar usando essa mesma aplicação, segue o link do repositório no Github.

A aplicação do To-Do List, desenvolvida durante o curso faz uso do Python 3.4 e Django 1.10, mas garanto a vocês que veremos como fazer o Deploy também na versão mais recente do Django e ficarão boquiaberto com a simplicidade do processo, claro que na versão 2.0, o Django sofreu algumas mudanças incompatíveis com versões anteriores, portanto, alguns cuidados com o desenvolvimento deverão ser observados ou caso queira portar a aplicação usando esta versão, você deverá ter os devidos cuidados.

Caso queiram dá uma olhada na aplicação no Heroku já em funcionamento, segue o link: http://easytaskapp.herokuapp.com/.

Tem um print da tela também. Caso esteja com preguiça de clicar no link acima.

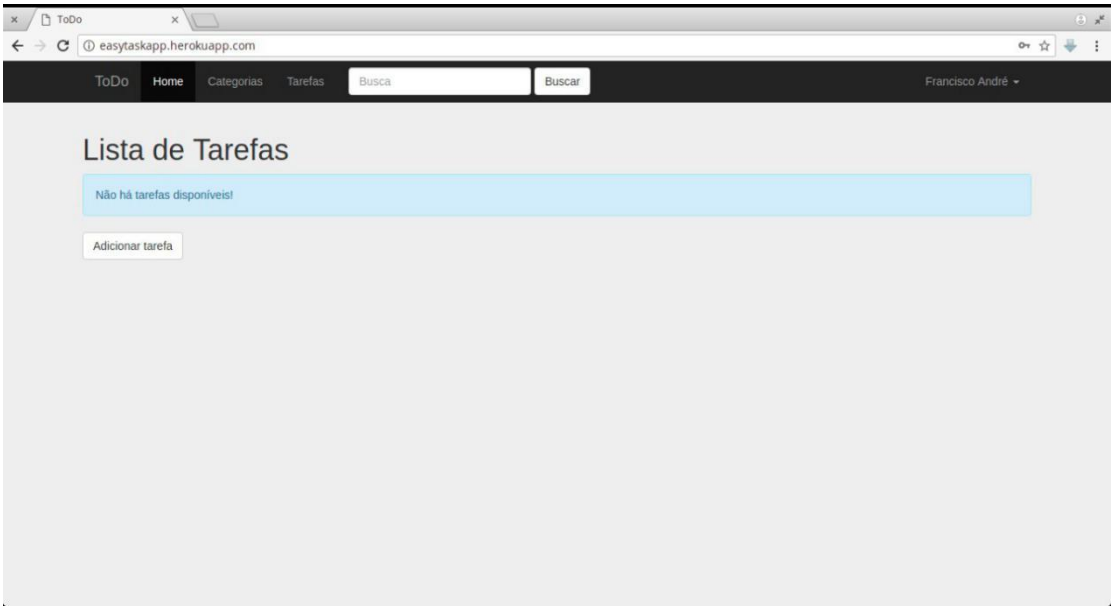

Figura 1 - Tela inicial da aplicação EasyTaskApp

Como o Heroku funciona

Seguinte, o funcionamento do Heroku é muito simples, por isso é tão fácil de usar. Basicamente o Heroku tem *Dynos*, que nada mais são do que *containers* unix isolados, é dentro desses *Dynos* que sua aplicação vai residir. Os *Dynos*, trocando em miúdos são também processos do sistema operacional, permitindo assim escalar sua aplicação rapidamente caso seja necessário.

Na versão free, a coisa funciona da seguinte maneira: você tem direito a um *Dyno* e se for uma conta verificada, você tem 1000 horas desse *Dyno* por mês. Caso não seja uma conta verificada, você vai ter 550 horas disponíveis.

Para verificar uma conta basta adicionar um cartão de crédito. Para isso, clique na sua foto do perfil, depois clique em *Account settings*, em seguida na aba *Billing*, depois no botão *Add Credit Card*, pronto, é só preencher o formulário e *voialá*, tudo verificado.

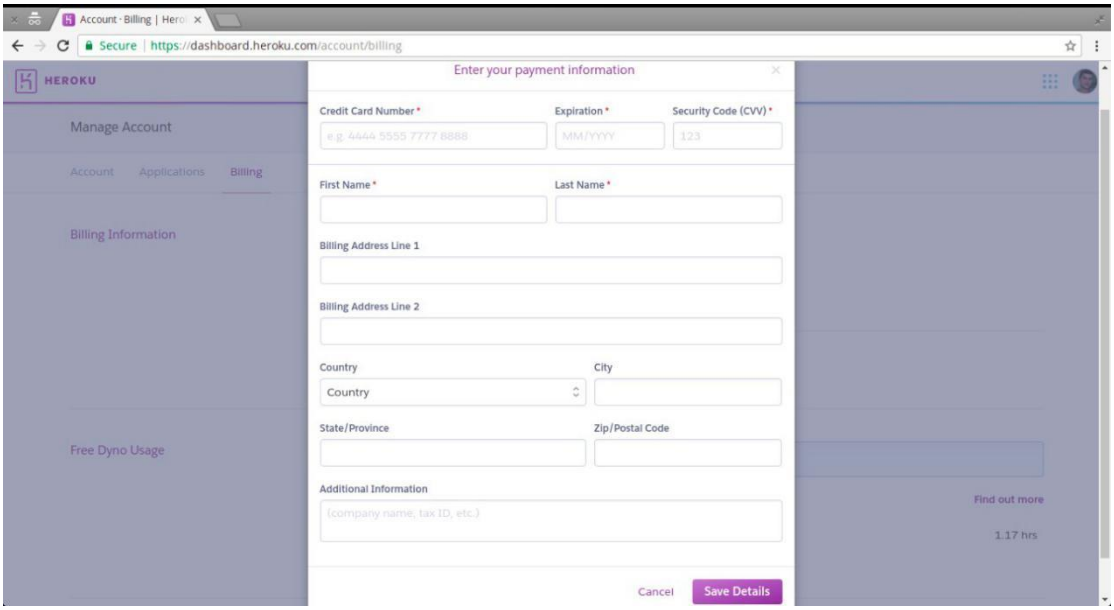

Figura 2 - Account Settings - Billing - Heroku

As limitações para o plano free vão desde a soneca que o *Dyno* tira depois de 30 minutos de inatividade, até o limite de 5 apps na plataforma. Para o plano free, mas com a conta verificada é possível adicionar alguns *add-ons,* que são softwares de complementos que servem para integrações com outros serviços. O único *add-on* que é permitido usar e que está disponível por *default* é o de conexão com o PostgreSQL.

Com todas essas limitações, o Heroku ainda é uma boa opção, tanto pela facilidade de uso quanto pela oferta, mesmo no plano free. Ideal para MVP - Produto Mínimo Viável, sigla em Inglês ou para quem está iniciando em Cloud Computing.

É importante salientar que toda vez que sua aplicação for acessada após um longo período de inatividade, 30 minutos para ser mais exato, como o *Dyno* vai está tirando aquele cochilo, vai demorar um pouco, coisa de alguns segundos, depois disso tudo flui normalmente.

Bom, acho que é o suficiente, o próximo passo é os preparos iniciais para o *Deploy*. Segue quente!

Criação de uma conta no Heroku

Bom, antes de tudo é preciso que você tenha uma conta no Heroku, então clique no link https://signup.heroku.com/ e forneça os dados solicitados, é simples, rápido e fácil.

Após criação da conta, vamos precisar executar a instalação do Heroku CLI, é um software de linha de comando que permite interagir com a plataforma totalmente a partir do terminal ou console do seu sistema operacional, após a instalação vamos ter uma *rapidinha* sobre os comandos principais que é disponibilizado. Segue o link de download do Heroku CLI.

https://devcenter.heroku.com/articles/heroku-cli#download-and-install

Cara, não tem segredo a instalação, é só fazer exatamente o que a página descreve ou seguir o processo de instalação da tua plataforma.

Beleza, feito a instalação você pode verificar se tudo saiu como o esperado digitando o seguinte comando no terminal:

```
heroku --version
```

A saída desse comando deverá ser algo parecido com o abaixo, claro que isso pode ser um pouco diferente dependendo da sua plataforma.

```
heroku-cli/6.16.3-4239951 (linux-x64) node-v9.9.0
```

É importante observar que todos os comandos do Heroku para que sejam interpretados pelo CLI (Command Line Interface), deverão ser precedidos pela palavra `heroku`, para ver uma lista com descrição dos comandos, apenas digite `heroku --help` no terminal. Para obter detalhes de um comando, use a sintaxe `heroku help COMMAND`.

Enfim, a primeira a coisa a ser feita após a instalação e verificação é fazer login no Heroku, como disse anteriormente, é possível administrar sua aplicação totalmente pelo terminal, veremos alguns comandos ao longo desse e-book.

Para fazer login, digite:

```
heroku login
```

Informe suas credenciais e se estiverem corretas, você receberá uma mensagem de sucesso.

Feito login, agora você pode executar comandos do seu terminal diretamente na plataforma, comandos para criação de uma nova app, comandos para administração do PostgreSQL e por aí vai, sugiro e recomendo você dar uma explorada nesse assunto.

Bom, como prometi, vou mostrar alguns desses comandos aqui.

Para listar suas aplicações:

```
heroku apps
```

Para criar uma app nova:

```
heroku apps:create APP_NAME
```

Se o parâmetro APP_NAME for omitido, o Heroku vai definir um nome de forma aleatória para sua aplicação.

Existe também um *alias* para o comando acima, que é o `heroku create`, digitando somente isso, irá executar exatamente o mesmo comando anterior, você pode também passar o nome da app quando estiver usando o alias.

Um detalhe importante a ser observado é que o nome da sua app precisa ser único, ou seja, o nome dela não deve existir de forma alguma na plataforma.

Continuando... Para verificar se a app está disponível após a criação, digite o comando abaixo:

```
heroku open
```

Você provavelmente não obterá um resultado “palpável”, podemos dizer assim, não fez ainda o *Deploy* de nenhuma aplicação.

Caso o comando anterior retorne um erro, você deve especificar a app, veja o comando abaixo.

```
heroku open --app APP_NAME
```

O seu navegador padrão vai ser aberto com a URL definida pelo Heroku na hora da criação da app, caso você tenha dado o nome da app na hora da criação, será esse nome seguido do `.herokuapp.com`, ou seja, será um subdomínio.

E por padrão, todas as aplicações serão um subdomínio de `herokuapp.com`.

Suave na nave? Até aqui tá de boa? Se tiver alguma bronca me manda um e-mail que terei o maior prazer em te ajudar.

Alguns dos meus contatos:

E-mail: ebook@django.school

É só informar qual e-book e a dúvida, blz?

Se for para tratar de outros assuntos, como aulas online e/ou presenciais (para aulas presenciais ver localização antes) individuais ou coletivas mande um e-mail para `francisco@django.school`.

Se quiser e puder claro, ajudar o autor do e-book, dê uma olhada nos meus cursos na Udemy, além de aprender você colabora a manter este e outros projetos gratuitos que estão por vir. Não se esqueça de usar o código **`CUPOM-19.99`** para obter o menor preço permitido pela plataforma do curso.

Se quiser algo específico sobre Django e Heroku por exemplo, tenho o Curso Desenvolvimento Web com Python e Django - Do zero ao Deploy.

Ok, fim da pausa, vamos continuar aqui com o assunto que interessa.

Para obter informações de uma app, use o comando:

```
heroku apps:info
```

Isso se você estiver dentro do diretório da app, se estiver em qualquer lugar no terminal e desejar ver as informações de uma app digite:

```
heroku apps:info --app APP_NAME
```

Em ambos os casos, você deve estar logado no Heroku.

Bom, você criou várias apps para testar e aprender sobre o Heroku, agora quer criar uma app pra valer e quando foi fazer isso viu que tinha atingido o limite de 5 apps, isso se você estiver usando a conta free. Não tem problema, você além de tudo isso que já fizemos, pode excluir uma aplicação, e como tudo no Heroku, de forma muito simples.

```
heroku apps:destroy APP_NAME -c APP_NAME
```

Atenção: isso vai excluir absolutamente tudo da sua aplicação e até onde eu sei, é irreversível, portanto, tenha certeza e muito cuidado ao executar esse comando, sem chororô depois e não diga que não avisei! ¯_(ツ)_/¯

Ainda vamos utilizar outros comandos ao longo de nossa jornada, por enquanto isso é o suficiente, vamos seguir em frente.

Preparação do projeto Django

Para que você consiga fazer o Deploy de uma aplicação no Heroku, há a necessidade de seguir algumas exigências, vamos a elas.

Atualmente, para a versão 1.9/1.10 do Django (acredito que a 1.11 também, mas não testei com ela, mais na frente vou mostrar com a 2.0), usada no projeto que vamos utilizar e disponível no Github, o Heroku espera alguns arquivos no projeto Django para que seja feito o Deploy, são eles: `Procfile, runtime.txt` e o arquivo `requirements.txt`. Parafraseando o senhor, "em verdade vos digo", o arquivo `Procfile` não é obrigatório, para um simples Deploy, caso haja necessidade de ajustes maiores na sua aplicação, aí sim, é obrigatório.

Não é obrigatório porque o Heroku detecta qual linguagem você está usando durante o processo e daí automaticamente cria e inicializa o servidor web adequado. Mas como diz o poema `Zen of Python`, explícito é melhor do que implícito! Por isso, vamos criar o nosso.

Uma coisa importante, esses arquivos precisam existir exatamente com este nome e na raiz do projeto, se for qualquer coisa diferente, nem que seja a extensão, não vai dar certo. Estes arquivos são também um simples arquivo de texto e através deles podemos declarar várias configurações para a nossa aplicação.

Então vamos lá, crie na raiz do seu projeto Django um arquivo chamado `Procfile`, exatamente da mesma forma, sem mais ou menos, no interior do arquivo digite a seguinte linha.

```
web: gunicorn todo.wsgi --log-file -
```

Vamos analisar cada item desse e vê quem é quem no jogo do bicho.

O camarada *web* é um *process type*, ou seja, é o processo que vai rodar, o "nome" para o seu comando, que nesse caso, é chamado de `web`. É importante observar que este cara é o único que vai receber requisições HTTP dentro do Heroku.

Continuação: `gunicorn` é nosso *web server*, é ele quem vai servir nossa aplicação nos servidores do Heroku. O que vem depois, `todo.wsgi` é a indicação de qual módulo Python é responsável por fazer essa integração entre a aplicação Django e o servidor web, no caso `gunicorn`, a primeira parte antes do **"."** é o diretório do

projeto, que é definido no momento da criação do projeto com o comando `django-admin`.

O último, mas não menos importante é o parâmetro `--log-file -`, esse cara serve para explicitar os logs da aplicação gerados pelo `gunicorn`, nesse caso eles serão exibidos na saída padrão, que é o terminal.

Após a criação do arquivo `Procfile`, vamos fazer agora o `requirements.txt`, se você estiver trabalhando em equipe, esse arquivo com certeza já foi criado, pois é a partir dele que será feita a instalação de todas as dependências do projeto, inclusive o Django. A sintaxe básica desse arquivo é: `lib==version`, por exemplo:

```
Django==1.10
```

Você pode instalar todas as dependências normalmente com o `pip`, do jeito que já faz, no final você pode usar o parâmetro `freeze` para criar e inserir todas as dependências dentro do arquivo.

```
pip freeze > requirements.txt
```

Vamos agora criar o arquivo `runtime.txt`, neste arquivo irá conter a versão do `Python` que queremos usar em nossa aplicação.

```
python-3.6.2
```

Caso queira uma versão diferente, consulte a documentação do Heroku para verificar quais versões estão disponíveis. Você pode consultar a documentação em: https://devcenter.heroku.com/articles/python-runtimes

Pronto.

Atualmente temos outra *lib* bem interessante que faz toda a gestão do ambiente virtual e das dependências, o `Pipenv`, o Heroku já suporta também essa nova *lib*, inclusive ela foi recomendada oficialmente pela comunidade Python.

Bom, estamos quase pronto, agora só precisamos instalar algumas dependências e fazer algumas configurações, aí estaremos prontos para o Deploy.

```
pip install gunicorn
pip install Unipath
pip install python-decouple
pip install dj-database-url
pip install whitenoise
```

Todas essas libs devem ser adicionadas ao `requitements.txt`, como já foi dito anteriormente, você só precisa digitar o comando abaixo.

```
pip freeze > requirements.txt
```

As dependências instaladas com os comandos acima, além de algumas serem "obrigatórias", compõe um grupo de boas práticas que seguem a recomendação do `12factors app`, portanto, usá-las irá garantir um Deploy sem dor.

Está chegando o grande momento!!! Vamos agora fazer a configuração de algumas *cositas* no projeto *e-pronto!!!* Essas configurações devem ser feitas no arquivo `settings.py` e `wsgi.py`.

No `wsgi.py` a coisa é bem simples, alteraremos apenas duas linhas.

```
from whitenoise.django import DjangoWhiteNoise
application = DjangoWhiteNoise(application)
```

Uma coisa importante a ser observada é que a linha `application = DjangoWhiteNoise(application)` deve vir depois da `application = get_wsgi_application()`.

Para o settings.py tem algumas coisinhas a mais, vamos a elas.

```
from decouple import config
from unipath import Path
from dj_database_url import parse as db_url

BASE_DIR = Path(__file__).parent
SECRET_KEY = config('SECRET_KEY')
DEBUG = config('DEBUG', default=False, cast=bool)
ALLOWED_HOSTS = ['.herokuapp.com']
DATABASES = {
     'default': config(
         'DATABASE_URL', default='sqlite:///' +
BASE_DIR.child('db.sqlite3'), cast=db_url
    )
}
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATIC_URL = '/static/'
```

Bom, agora sim, estamos prontos, o bom é que essas configurações só serão feitas uma vez e com versões mais recentes do Django, a 2.0 para ser mais exato, é extremamente simples, daqui a pouco mostro como fazer com a versão mais nova.

Configurações sensíveis

Antes de continuarmos, vou abrir um parênteses para falar da lib `python-decouple`, além de ser uma excelente ferramenta e ter como autor um cara que tenho grande apreço, chamado Henrique Bastos, uma pessoa fantástica, essa lib nos permite separar todas as configurações sensíveis do projeto, como dados de acesso ao banco, configurações de envio de e-mails, chaves de encriptação enfim.

Para que você consiga fazer essa separação, você precisa criar um arquivo chamado *.env* ou um arquivo *.ini*, no arquivo vai todas as configurações sensíveis e geralmente eles devem ser ignorados pelo `git`. Outra coisa importante que deve ser observada é a localização do arquivo, ele deve existir na raiz do projeto.

Eu particularmente prefiro usar o *.env*, nas referências tem o link da documentação e lá tem mais detalhes. *Let's go!*

Deploy

A essa altura você já deve ter o `git` instalado e configurado na sua máquina, se não tem, acesse o site https://git-scm.com/ e instale de acordo com sua plataforma, para o Linux Ubuntu ou seus derivados ou mesmo Debian, pode usar o comando abaixo.

```
sudo apt update && sudo apt install git git-core -y
```

Isso para quem está na versão 16.04 ou superior do Ubuntu. Para versões anteriores use o comando seguinte:

```
sudo apt-get update && sudo apt-get install git git-core -y
```

Depois você precisa *setar* algumas informações do usuário no git.

```
git config --global user.name “Seu Nome”
git config --global user.email “seu@email.com”
```

Fechou.

Não esqueça de criar o arquivo `.gitignore` e adicione nele o que você gostaria ou não precisa enviar para o Deploy ou repositório do projeto.

Recapitulando, para a efetivação do Deploy, claro, depois de ter criado os arquivos `Procfile, runtime.txt, requirements.txt` e ter feito as instalações e configurações necessárias precisaremos dos comandos abaixo:

Observação: Alguns desses comandos só serão executados na primeira vez do Deploy, nas demais, não será mais necessário.

```
heroku create <APP_NAME>*
git remote add heroku <URL>*
git add .gitignore*
git commit -m “Ignorando o que não é do projeto.”
git add .
git commit -m “Subindo projeto.”
heroku config:set SECRET_KEY=AJHGK87653**
git push heroku master
```

Pronto, isso é o suficiente para o primeiro Deploy, claro que agora vamos precisar executar as `migrations`. Para qualquer comando do Django que você deseja executar, deverá usar o `heroku run` precedendo o comando, como em:

```
heroku run python manage.py migrate
```

Os comandos marcados com o “*” são comandos que geralmente só é preciso no primeiro Deploy.

O comando marcado com “**” é necessário também no primeiro Deploy ou se você desejar atualizar a chave SECRET_KEY que é a chave usada pelo Django para validar uma série de coisas na sua aplicação. Um detalhe super importante é que se você *setar* essa variável pelo terminal com o comando acima, você não poderá usar na sua chave o caractere alguns caracteres especiais que tenham um significado importante para o `bash`, portanto, fique atento.

Caso queira usar e é altamente recomendado usar uma chave mais segura, eu aconselho a criar essa variável de ambiente usando o seu Dashboard, basta acessar sua conta no Heroku e ir até as configurações da aplicação.

O parâmetro <APP_NAME> e <URL>, são respectivamente o nome da aplicação no Heroku caso você deseje informar, e a URL é gerada automaticamente pelo Heroku logo após o primeiro comando, o de criação da aplicação.

Se desejar criar por exemplo, um super usuário para a acessar a administração do Django, como disse alguns parágrafos acima, basta executar com o comando abaixo,

mais uma vez, qualquer comando do Django que deseje executar, basta preceder com o comando com o `heroku run`.

```
heroku run python manage.py createsuperuser
```

Informe as credenciais como solicitado no `shell` e pronto, tudo feito.

Só para deixar claro, a partir do segundo Deploy, você só vai precisar dos comandos abaixo, que nada mais são comandos do Git.

```
git add .
git commit -m “Sua mensagem.”
git push heroku master
```

Isso é tudo!

Observação: é boa prática fazer uso de processos de DevOps para o Deploy, infelizmente não é o foco desse e-book, é assunto suficiente para um novo livro, quem sabe no futuro produza um com esse tema!

Deploy com Django 2.0

Felizmente, para a versão 2.0 do Django, esse processo ficou extremamente simples, foi removido a necessidade de várias configurações, se limitando a apenas alguns passos, como será mostrado logo a seguir.

Levando em consideração que você já tenha a APP criada no Heroku e que esteja com seu projeto Django criado, você só precisará instalar algumas dependências e criar os arquivos `Procfile` e `requirements.txt`.

```
#Procfile
web: gunicorn djangoproject.wsgi --log-file -
```

```
pip install gunicorn
pip install django-heroku
```

```
#requirements.txt
pip freeze > requirements.txt
```

```
#settings.py
import django_heroku
django_heroku.settings(locals()) # no final do arquivo
```

Importante: todos os passos leva em consideração que você já esteja com o ambiente virtual criado e com o Django instalado, isso quer dizer que dentro do arquivo `requirements.txt` deve haver uma linha referente ao Django como dependência. Para o Django 2.0 as configurações do SECRET_KEY, DEBUG, ALLOWED_HOSTS, STATIC_URL, STATIC_ROOT e etc, também devem ser mantidas, de preferência separadas do arquivo `settings.py`.

Conclusão

Python e Django são tecnologias fantásticas, atualmente essas duas tecnologias têm sido minhas *stack* principal de trabalho. O Heroku é simplesmente fantástico, uma paixão a parte, o pessoal lá realmente fez um excelente trabalho.

Este e-book não tem o propósito de esgotar o assunto Deploy de uma aplicação Django no Heroku, o objetivo principal é facilitar o entendimento e mostrar o quão fácil é esse processo usando essas tecnologias.

Me dediquei bastante para escrever um e-book que fosse de fácil leitura e compreensível, mas ainda assim posso ter deixado as coisas ainda mais complicadas ao invés de esclarecer tudo, mas se isso aconteceu por favor, não hesite em me mandar uma mensagem, aproveita e envie também feedback, isso é muito importante, pode ser elogios, críticas, dicas ou agradecimentos, na verdade, qualquer coisa. Do mesmo jeito se cometi algum crime contra a nossa gramática eu também aceitarei com o maior prazer a sua correção, claro que algumas palavras foram escritas como são usadas no nosso dia a dia, portanto, foram escritas de forma proposital.

No mais, me despeço nesse momento desejando a todos sucesso e prosperidade em todas as áreas de suas vidas.

Um forte abraço e fui!!!

Referências

Bom, para escrever este e-book eu tive que consultar principalmente minha “cuca” e a documentação do Heroku, Gunicorn e Django, segue os links abaixo. Na verdade, eu costumo dizer que a melhor amiga do programador e que para os crentes, o manual do mundo é a Bíblia, para os programadores (ou pelo menos alguns), além da Bíblia, claro, a documentação.

Heroku dev center

https://devcenter.heroku.com/

Gunicorn

http://gunicorn.org/

Django

https://www.djangoproject.com/

Outras documentações consultadas e recomendadas:

https://pypi.org/project/dj-database-url/

https://github.com/mikeorr/Unipath

http://whitenoise.evans.io/en/stable/

https://pypi.org/project/django-heroku/

https://github.com/henriquebastos/python-decouple